# PHP7

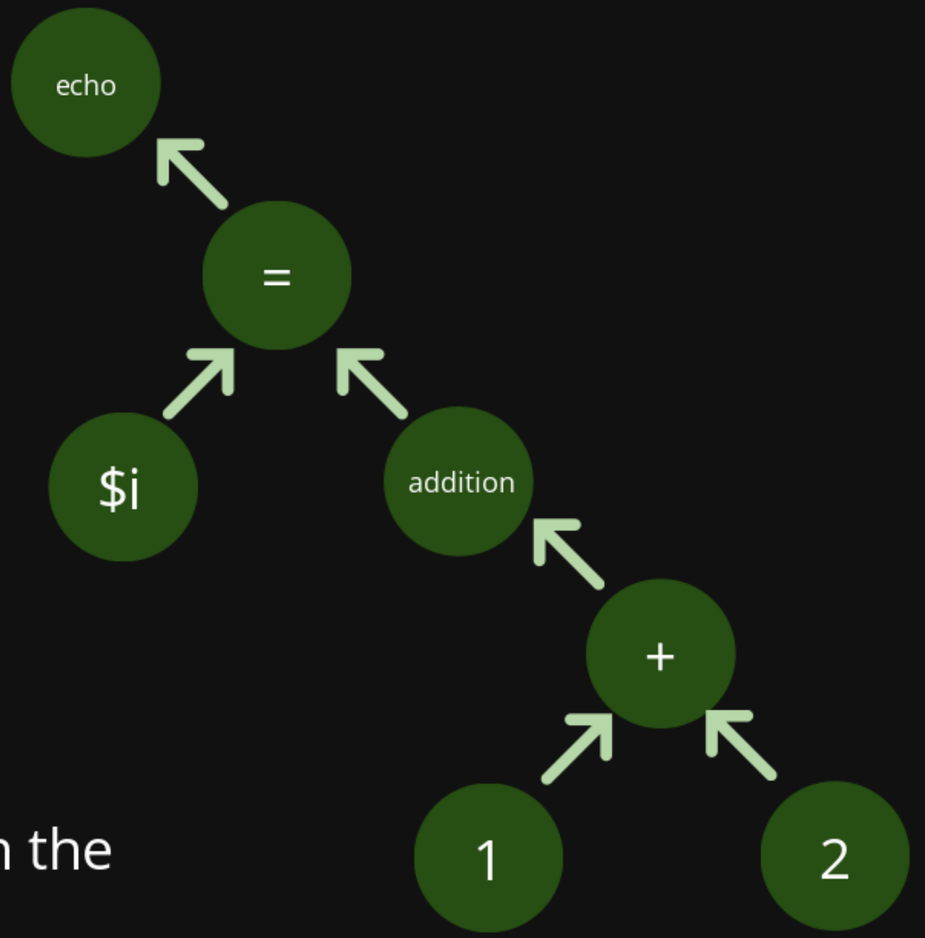
and the RFCs

# Technical Interpreter Changes

- Abstract Syntax Tree
- jsond
- Fast Parameter API
- Native TLS Support
- NG merged to Master
- 64 bit platform improvements for string length and integers
- gc_collect_cycle as function pointer

# Abstract Syntax Tree

```php
<?php
function addition($a, $b){
    return $a + $b;
}
$i = addition(1,2);
echo $i;
```




Atomic operations are harder to predict and to analyze before execution

AST

- higher memory allocation
- allows optimizations based on the graphs
- foundation for JIT improvements

# JSOND

# Opensource

Mainly Internal implications

C headers

C Macros

# Fast Parameter Parsing API

Reduces the overhead of repeated parsing
of character to gain a performance boost

NFA to DFA

instead of passing a character to determine if what
type it uses C-Macros

# Native TLS Support

## Thread-local Storage

Mainly for Better thread handling and stability

# NG - Merge to Master

Complete change of execution Model

Foundation for future JIT implementations

# 64Bit improvements for string length and integers

Implementation of platform 64 bit handlers for a more consistent behaviour.

# Garbage Collection Cycles as a function Pointer

For future Extensions a GC collection function pointer is added to hook into the cycle

# Error Behaviour

- Engine and Parse Exceptions
- Reclassify E_STRICT Notices
- Catchable "call to member function of a non object"

# Engine and Parse Exceptions

- 2 new "core" Exceptions (ParseException & EngineException)
- The error model changes to an exception model!
- Exceptions can be handled gracefully
- final blocks are invoked
- __destruct() is invoked
- eval() throws an exception

# Reclassify E_STRICT notices

- It will mainly converted to a E_NOTICE and loose any other meaning

# Catchable "call to member function of a non object"

- `E_FATAL_ERROR will be converted to a E_RECOVERABLE_ERROR`

# Cleanup

- Constructor behaviour of internal classes
- Remove PHP4 Constructor
- Remove date.timezone warning
- Preserve fractional part in json encode
- Remove alternative PHP tags
- Fix handling of custom session handler return value
- Remove HEX support in numeric strings
- Removal of dead SAPIS

# Constructor behaviour of internal classes

These Classes will now ALL throw an Exception if the constructor "fails"

- finfo
- PDO
- Collator
- IntlDateFormatter
- MessageFormatter
- NumberFormatter
- ResourceBundle
- IntlRuleBasedBreakIterator
- UConverter
- SplFixedArray
- ReflectionFunction
- ReflectionParameter
- ReflectionMethod
- ReflectionProperty
- ReflectionExtension
- ReflectionZendExtension
- Phar
- PharData
- PharFileInfo

# PHP 4 Constructor removed

Finally this wont be called as a constructor anymore

```php
<?php

class Test {
   function Test() {
        echo 'test';
   }
}
```

# Switch default multiples

no more multiple defaults

it will cause a compile error!

```php
<?php
switch ($expr) {
  default:
    neverExecuted();
    break;
  default:
    executed();
}
```

# Function behaviour

- filtered unserialize
- generator return expresssions
- continue output buffering despite aborted connection

# filtered unserialize

security measure to prevent code
injections via classes in serialized strings

```php
<?php
// this will unserialize everything as before
$data = unserialize($foo);
// this will convert all objects into __PHP_Incomplete_Class object
$data = unserialize($foo, ["allowed_classes" => false]);
// this will convert all objects except ones of
// MyClass and MyClass2 into __PHP_Incomplete_Class object
$data = unserialize($foo, ["allowed_classes" => ["MyClass", "MyClass2"]);
//accept all classes as in default
$data = unserialize($foo, ["allowed_classes" => true]);
```

# Continued Buffering even on connection los

Basically you can use the output buffer -> ob_start(callback)
it will remain as long as the buffer layer remains

# Interpreter Behaviour

- Context Sensitive Lexer
- Uniform variable syntax
- ZPP Failure on Overflow
- Integer Semantics
- Unicode Codepoint Escape Syntax
- Fixed foreach behaviour
- Fixed list behaviour inconsistency

# New reserved type names!

the following types are reserved in php7 and can not be used as interface or trait  names

- int
- float
- bool
- string
- true, false
- null

# Context Sensitive Lexer

Lexems are put in to Context so the interpreter
commando list != ClassName->list

```php
<?php
class ContentSensitiveLexer {

    /**
     * an example for an error
     * -> const class
     */

    /**
     * @var array
     */
    public $list = [];

    const continue = 12;

    const array = 10;
}

echo ContentSensitiveLexer::continue; // is 12
echo ContentSensitiveLexer::array; // is 10
```

# Uniform variable syntax

it's for a clearer context in the nesting ***hierarchy***

WILL BREAK NON EXPLICIT DYNAMIC VARIABLE GENERATION

Incompatiblities

```php
<?php
                        // old meaning              // new meaning
$$foo['bar']['baz']     ${$foo['bar']['baz']}     ($$foo)['bar']['baz']
$foo->$bar['baz']       $foo->{$bar['baz']}       ($foo->$bar)['baz']
$foo->$bar['baz']()     $foo->{$bar['baz']}()     ($foo->$bar)['baz']()
Foo::$bar['baz']()      Foo::{$bar['baz']}()      (Foo::$bar)['baz']()
```

# ZPP failure on overflow

fixes float overflows! ;)

# Integer Semantics

```php
<?php
// Was: int(-9223372036854775808)
// Now: int(0)
var_dump((int)NAN);
// Was: int(-9223372036854775808)
// Now: int(0)
var_dump((int)INF);
// Was: int(4611686018427387904)
// Now: bool(false) and E_WARNING
var_dump(1 << -2);
// Was: int(8)
// Now: int(0)
var_dump(8 >> 64);
```

# Unicode Codepoint Escape Syntax

possibility to determinate the difference between

```
echo "ma\u{00F1}ana"; // pre-composed character
echo "man\u{0303}ana"; // "n" with combining ~ character (U+0303)
```

which are not visible outside the unicode escape syntax

```
echo "mañana";
echo "mañana";
```

and allows access to other symbols within the unicode

```
echo "\u{1F602}"; // outputs 😂
```

# Fixed foreach behaviour

Inconsistencies for edge cases are fixed

```
$ php -r '$a = [1,2,3]; foreach($a as $v) {echo $v . " - " . current($a) . "\n";}'
1 - 1
2 - 1
3 - 1
```

# Fixed list behaviour

Strings now can be split by list

```
list($a,$b) = "str"; //will now be handles like $a = "s"; $b = "t";
```

I can has new featurez!